Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Информационные системы и базы данных

Акимов Максим Евгеньевич

Реализация пакета аналитических функций в PostgreSQL. Алгоритмы кластеризации.

Бакалаврская работа

Научный руководитель:
к. ф.-м. н., доцент Графеева Н.Г.

Рецензент:
ведущий инженер отдела автоматизации Матвеева И.Е.

Санкт-Петербург
2018

SAINT-PETERSBURG STATE UNIVERSITY
Mathematics & Mechanics Faculty
Informational Analytical Systems Chair

Akimov Maksim

PostgreSQL analytical functions package development. Clustering algorithms.

Bachelor's Thesis

Scientific supervisor:
Ph.D., Associate Professor Natalia Grafeeva

Reviewer:
leading engineer of automatisation Irina Matveeva

Saint Petersburg
2018

1. Оглавление

2. **Введение**

Информационная составляющая современного мира невероятна обширна. Практически все сферы деятельности человека, так или иначе, связаны с различными видами хранения и обработки данных. Сложно представить такие направления как: медицина, экономика, торговля и многие другие без понятия больших данных. Но сами по себе данные не несут в себе такой ценности как умение грамотно анализировать их. Аналитические методы позволяют человеку применять различные математические методы для выявления знаний из данных, и впоследствии использовать полученную информацию для оптимизации работы, выявления угроз и опасностей, а также выявления направлений в развитии бизнеса. Все это позволяет людям не только избежать огромных финансовых затрат, но и сократить временные и ресурсные аспекты решения возникающих проблем.

Так как наиболее распространенным способом хранения и обработки данных являются базы данных, то и большинство аналитических методов основано на обработке больших баз данных с целью нахождения, как заранее известных результатов, так и выявлением новых уникальных закономерностей. В связи с этим многие из основных производителей систем управления базами данных, такие как Oracle и Microsoft работают над внедрением в свои продукты пакетов аналитических функций и процедур, обладающих высоким спросом, что дает возможность каждому пользователю получить доступ ко многим мощным методам анализа данных сразу после установки. Однако не все СУБД обладают достаточным количеством ресурсов для разработки подобных программных решений. Одной из таких СУБД является PostgreSQL, активно развивающейся и все чаще в последнее время внедряемой многими российскими компаниями в качестве основного продукта для разработки баз данных.

Наряду с имеющимися недостатками, один из которых был упомянут выше, PostgreSQL обладает и рядом преимуществ, основным из которых является ее доступность, благодаря чему многие компании и аналитики предпочитают использовать в своей работе именно ее. Однако многие из них сталкиваются с невозможностью полностью реализовать свои потребности функционалом, предлагаемым после стандартной установки системы. В такие моменты и возникает потребность поиска сторонних решений или же смена среды разработки. Стоит отметить что на сегодняшний день существует ряд методов способных в какой-то мере решить возникающие проблемы, однако многие из них, для реализации поставленных задач, используют сторонние программные продукты, встраиваемые в PostgreSQL, что, в некоторых случаях может приводить к непредсказуемому поведению или возникновению конфликтных ситуаций. Кроме того, рядовому пользователю незнакомому с тем, как функционирует

вспомогательный продукт, может доставить ряд неудобств ознакомление с принципом его работы. В связи с вышесказанным возникает необходимость реализации пакета, способного приблизить PostgreSQL к коммерческим продуктам, используя при этом лишь средства, предоставленные самой СУБД.

3. Постановка задач

Проект написания пакета аналитических функций является многосторонним и работу над ним вела команда разработчиков. Однако в данном тексте мы осветим лишь одну из идей проекта. Целью данной работы является создание пакета аналитических методов, включающего в себя наиболее популярные алгоритмы кластеризации, а также способы оценки ее качества.

Для достижения этой цели были сформулированы следующие задачи:

- выполнить обзор существующих методов решения данной проблемы
- провести анализ алгоритмов кластеризации и выбрать наиболее актуальные из них
-провести анализ методов оценки качества кластеризации и выбрать наиболее подходящие
- реализовать пакет методов кластеризации и индексов оценки качества средствами PostgreSQL

4. Обзор существующих решений

На сегодняшний день существует множество подходов, позволяющих использовать аналитические методы в PostgreSQL. В этой главе будут описаны наиболее популярные из них.

Одним из наиболее крупных проектов, связанных с добавлением в PostgreSQL аналитических методов является MADlib [1] – свободно распространяемая библиотека с открытым исходным кодом, разработанная компанией Apache. Данная библиотека обеспечивает параллельные реализации математических, статистических и машинных методов обучения для структурированных и неструктурированных данных. MADlib позиционируется как продукт подходящий для решения вопросов связанных с классификацией, регрессией, кластеризацией, описательной статистикой, поиском ассоциативных правил и моделированием тем. Наиболее интересным среди данных вопросов для нас является кластеризация, которая в данном продукте представлена лишь алгоритмом K-means. Архитектурная составляющая данного продукта состоит из трех основных компонентов:

- Python driver functions. Эти функции являются основной входной точкой пользовательского ввода и в значительной степени отвечают за управление потоком алгоритмов.

- C++ implementation functions. Эти функции являются C++ определением основных функций и агрегатов, требуемых для конкретных алгоритмов.

- уровень абстракции базы данных C++. Эти функции служат для предоставления программного интерфейса, который абстрагирует всю внутреннюю информацию PostgreSQL и обеспечивает механизм, при котором MADlib может поддерживать различные backend-платформы и сосредоточиться на внутренней функциональности, а не на логике интеграции платформы.

Схематичное изображение вышеописанной архитектуры, представленное в Apache MADlib documentation [1] показано на рисунке 1.

Более узкоспециализированным, однако не менее функциональным является расширение PostGIS [2], предоставляющее в том числе и возможность использовать алгоритмы кластеризации. PostGIS предоставляет поддержку пространственных и географических объектов в PostgreSQL, путем добавления дополнительных типов. Кроме этого расширение добавляет новые функции и индексы, которые применяются к этим типам. Все это происходит на основе построения библиотеки C, которую PostgreSQL может загружать в PostgreSQL backend, после чего происходит связь функций и структур из этой библиотеки с типами и функциями языка SQL. PostGIS является open source проектом и распространяется бесплатно. Основными преимуществами данного программного продукта является предоставление пользователю возможности использовать такие концепции как: Обработка и аналитические функции как для векторных, так и для растровых данных для сплайсинга, обработки, преобразования, переклассификации и сбора или объединения с помощью SQL, алгебра растрового отображения для мелкозернистой растровой обработки, пространственные функции перепрограммирования SQL для векторных и растровых данных, поддержка топологии сетей и многие другие.

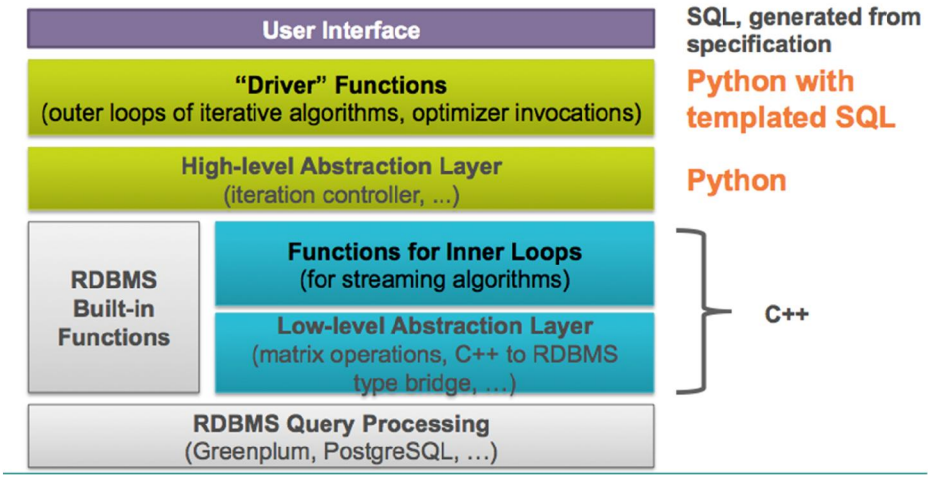

Рисунок 1. Архитектура MADlib

Еще одним подходом к интеграции аналитических методов в PostgreSQL является использования PL/R [3], разработанного Джо Конвэем. PL/R – это расширение, которое позволяет писать функции PostgreSQL и агрегатные функции

на языке статистических вычислений R. Необходимым условием для использования данного расширения является установленная на том же сервере что и PostgreSQL среда языка R.

Обзор существующих решений приводит к мысли о том, что у всех ныне существующих продуктов имеется определенный ряд недостатков. Так, например, использование расширение PL/R может оказаться крайне нетривиальным на практике для рядового пользователя, ведь прежде чем начать его использование необходимо провести ряд действий, радикально отличающийся от стандартного сценария использования PostgreSQL, что может значительно усложнить работу с данными. Кроме того, дополнительная установка средств языка R может быть неуместна в некоторых случаях. Решения предлагаемы PostGIS и MADlib так же в своей основе используют сторонние программные продукты такие как функции, построенные на основе библиотек C, что подразумевает загрузку этих последних. Более того MADlib использует в своей архитектуре функции написанные на Python, о которых на официальной странице PostgreSQL [4] сказано, что PL/Python представлен только в виде «недоверенного» языка. Недоверенность языка ведет к тому, что действия пользователя не могут быть ограничены. Это значит, что только человек, реализовавший фикцию отвечает за то по какому сценарию эта функция может быть использована, так как она может делать все то, что доступно администратору базы данных. Доверенная вариация plpython может появиться в будущем, если в Python будет разработан безопасный механизм выполнения. Создавать функции на недоверенных языках, таких как plpythonu, разрешено только суперпользователям.

Стоит также отметить что одним лишь алгоритмом K-means, реализованным в MADlib, невозможно охватить все многообразие задач, возникающее в результате обработки данных, даже учитывая все преимущества данного алгоритма

5. Выбор алгоритмов для реализации

Одним из наиболее популярных неиерархических алгоритмов является K-means [5]. Данный алгоритм является хорошо масштабируемым для больших объемов данных, благодаря чему может быть использован во многих областях. Зачастую к его использованию прибегают в том случае, когда перед аналитиком предстает задача поиска групп в неразмеченном наборе данных.

Примером, случаев, в которых данный K-means может быть применим могут послужить следующий ситуации: Выявление ботов и аномалий, сортировка датчиков, поведенческая сегментация и многие другие.

K-means кластеризует данные путем распределения элементов исходной выборки по n группам равных дисперсий, минимизируя при этом параметр,

называемый внуктрикластерной суммой квадратов. Каждая из полученных таким способом групп именуется кластером и характеризуется единственным параметром – центроидом кластера. Центроиды не обязательно являются элементами исходной выборки, хотя и существуют в том же пространстве, что и все остальные элементы.

Подходя к описанию данного алгоритма более формальном стоит отметить, что необходимым условием его работы является задание параметра n - количества кластеров, после чего его работу можно разбить на три этапа:

- Первым этом является стартовая выборка центроидов.
- На втором шаге происходит инициализация каждого элемента исходного множества номером наименее удаленного от него кластера.
- Финальным этапом является создание новых центроидов, основанное на среднем значении всех элементов, вошедших в данный кластер на предыдущем шаге.

Последние два шага циклично выполняются до тех пор, пока расстояние между старым новым центроидами не станет меньше какой-то заранее определенной величины.

K-medians является одной из вариаций алгоритма K-means, в котором вместо средних расстояний вычисляются медианы. Такой подход позволяет уменьшить влияние выбросов на работу алгоритма и получить более компактные кластеры нежели кластеры получаемы в K-means. Принцип работы данного алгоритма аналогичен работе K-means за исключением подсчета центроидов.

Алгоритм DBSCAN [6] один из плотностных алгоритмов, рассматривающих кластеры как области высокой плотности, разделенные областями с низкой плотностью, в связи с чем кластеры, найденные DBSCAN, могут быть любой формы, в отличие от k-means, результатом которого являются выпуклые фигуры. Центральным компонентом DBSCAN является концепция ключевых элементов, которые находятся в областях с высокой плотностью. Таким образом, кластер представляет собой набор ключевых элементов, каждый из которых расположен близко друг к другу (измеряется с помощью некоторой меры измерения расстояния) и набор неосновных образцов, которые близки к ключевым элементам, но сами не являются ключевыми. Для алгоритма есть два параметра: минимальное количество соседей и радиус окрестности точки, которые формально определяют, что мы имеем в виду, когда говорим плотным. Более высокое минимальное количество соседей или меньший радиус окрестности указывают более высокую плотность, необходимую для формирования кластера.

Более формально мы определяем ключевой элемент как образец в наборе данных, такой что на заданном от него расстоянии находиться минимальное заданное количество элементов, которые определяются как соседи ключевых элементов. То есть, что образец ядра находится в плотной области векторного

пространства. Кластер представляет собой набор базовых образцов, которые могут быть построены путем рекурсивного взятия образца ядра, поиска всех его соседей, которые являются базовыми образцами, нахождением всех их соседей, которые являются образцами ядра, и так далее. Кластер также имеет набор непрофильных выборок, которые являются соседями ключевых элементов в кластере, но сами таковым не являются. Интуитивно эти образцы находятся на периферии кластера.

Любая базовая выборка является частью кластера по определению. Любой образец, который не является образцом ядра, и является, по меньшей мере, на расстоянии eps от любого образца ядра, считается внешним.

FOREL - алгоритм кластеризации, предложенный в работе [7], в котором объекты объединяются в кластеры в местах где концентрация их наибольшая. Для работы алгоритма необходимо задать параметр r, который является радиусом, после этого все точки, находящиеся на расстоянии, не превышающем r помещаются в кластер. Следующим шагом является вычисление центра тяжести полученного множества точек или, другими ловами, поиск точки являющимся формальным элементом, вокруг которого образуется новое множество элементов, и процедура повторяется. Все это происходит до тех пор, пока движение центра тяжести не прекратиться, то есть пока центр тяжести не окажется в месте наибольшего сгущения элементов выборки. Точки, вошедшие в этот кластер, помечаются класетризованными и в дальнейшей разметке не участвуют. Таким образом данный метод локализует места, в которых плотность точек наибольшая и строит вокруг них кластеры.

Было бы упущением не упомянуть об алгоритмах иерархической кластеризации, результатом работы которых является не единственное конечное разбиение на кластеры, а система вложенных разбиений, называемая дендограммой. Иерархические алгоритмы по принцип работы делят на дивизимные и агломеративные, последние из которых являются более популярными.

Агломеративный [8] или восходящий алгоритм кластеризации на начальном шаге разбивает набор данных на количество кластеров, равное количеству элементов. Каждая последующая итерация объединяет пару наименее удаленных друг от друга кластеров в один. Алгоритм останавливается в момент, когда количество кластеров становиться равным 1. Основным параметром данного метода является выбор метрики. Выбор метрики определяет форму кластеров, так в одной метрике какие - то элементы могут быть близки друг к другу, в то время как использование другой метрики приведет к абсолютно противоположному результату. Чтобы найти оптимальное разбиение необходимо провести процесс кластеризации до конца и только после этого имеет смысл искать экстремальное значение целевой функции оценки качества кластеризации.

В результате стоит отметить, что каждый из рассмотренных методов имеет свои плюсы и минусы и нельзя однозначно определить лучший из них. Результат кластеризации во многом зависит от структуры начальных данных и в случаях где один алгоритм может показать отличный результат другой абсолютное не справиться с поставленной задачей. Стоит так же отметить, что до сей поры рассуждения о том насколько хорош тот или иной результат основывались лишь на визуальном его представлении, что определенно является неплохим методом оценки качества, однако возникает большое количество ситуаций, в которых данный подход не даст желаемого результата, поэтому возникает необходимость в альтернативных способах оценки кластеризации.

Информация о методах оценки качества кластеризации была описана в источнике [9] исходя из которого можно сделать вывод о том, что оценка качества кластеризации сильно усложняется в случае отсутствия кластерной структуры. Это факт так же неизменен в случае однокластерной структуры данных. С оценкой результата кластеризации K-means могут справиться многие алгоритмы, однако в случае DBScan ситуация несколько усложняется в связи с неоднозначностью результирующих форм кластеров. Для оценки результатов были рассмотрены метрики Silhouette, Dunn и DB.

Silhouette [9] – индекс силуэта. Данная метрика позволяет оценить качества кластеризации основываясь лишь на самой выборке. Коэффициент Силуэт рассчитывается для каждого элемента. Данный расчет основывается на разности двух величин: среднем расстоянии между объектом и всеми остальными объектами того же кластера, а также минимальным расстоянием между объектом и объектами ближайшего к данному кластера. Результат вычитания нормируется наибольшим значением, для приведения оценки к фиксированному диапазону. Итоговым значением Силуэта для всей выборки является усредненное значение результатов для каждого из элементов выборки.

$$Silh = \frac{1}{N}\sum_{i=1}^{N}\frac{\left(\min\limits_{l\neq k}\min\limits_{a_j\in c_l}\|a_i - a_j\|\right) - \left(\frac{1}{|c_k|-1}\sum_{i\neq j, a_i\in c_k, a_j\in c_k}\|a_i - a_j\|\right)}{\max\left(\frac{1}{|c_k|-1}\sum_{i\neq j, a_i\in c_k, a_j\in c_k}\|a_i - a_j\|, \min\limits_{l\neq k}\min\limits_{a_j\in c_l}\|a_i - a_j\|\right)}$$

Итоговое значение лежит в диапазоне от -1 до 1, чем больше полученный результат, тем лучшим является разбиение на кластеры. Таким образом данная метрика показывает разницу между средними внутрикластерным и межкластерными расстояниями.

Dunn [9] – индекс оценки качества кластеризованных данных основанная на сравнении размеров кластеров и расстояния между ними. Так результат метрики получается по формуле:

$$Dunn = \frac{\min\limits_{i \neq j} \min\limits_{a_{ik} \in c_i, a_{jl} \in c_j} \|a_{ik} - a_{jl}\|}{\max\limits_{i} \max\limits_{a_{ik} \in c_i, a_{il} \in c_i, i \neq l} \|a_{ik} - a_{il}\|}$$

Таким образом представляет собой отношение минимума расстояний между разными кластерами к максимуму расстояний между элементами одного кластера. В отличии от метрики силуэт является абсолютной метрикой, чем выше значение оценки, тем лучше структурированы кластеризованные данные.

DB – индекс Дэвиса Болдуина [14] определяет среднюю схожесть данного кластера с наименее отличным от него кластером, усредненное по всем кластерам. Исходя из этого наилучшем результатом кластеризации будет считаться наименьшее значение индекса DB. Вычисление индекса построено на отношении внутрикластерного рассеяния $S_i = (\frac{1}{T_i} \sum_{j=1}^{T_i} |X_j - A_i|^p)^{\frac{1}{p}}$, значение которого должно быть малым и расстояние между кластерами $M_{i,j} = (\sum_{k=1}^{n} |a_{k,i} - a_{k,j}|^p)^{\frac{1}{p}}$.

$$DB = \frac{1}{N} \sum_{i=1}^{N} \max_{j \neq i} \frac{S_i + S_j}{M_j}$$

Таким образом, чем меньше кластеры похожи друг на друга тем лучший результат метрики будет получен.

6. Реализация

В качестве инструмента для реализации выбранных аналитических методов было принято решение использовать процедурное расширение языка SQL - PL/pgSQL, которое используется в PostgreSQL. PL/pgSQL в отличии от традиционного SQL позволяет группировать последовательности запросов и вычислительные блоки внутри сервера базы данных, что в значительной мере увеличивает скорость обработки информации, так как данный подход решает такие проблемы как: излишнее взаимодействие клиента и сервера, передача промежуточных данных между исполнителями, значительно увеличивающие время выполнения запросов, а так же многократное выполнение одних и тех же запросов.

Была разработана схема, которую можно разворачивать (устанавливать) в базах данных PostgreSQL. В данном пакете реализован широкий спектр функций используемых для генерации наборов данных, расчета расстояний и т.п. Однако в данной работе мы сделаем акцент на описание работы методов кластеризации и метрик оценивания результатов кластеризации. Далее приведено описание реализации этих алгоритмов в рамках схемы anfun.

6.1. Развертывание схемы

Для установки схемы пользователь должен обладать правом на создание схемы. Чтобы предоставить пользователю это право используется команда:

```
GRANT CREATE ON DATABASE <database name> TO <user role>
```

Для использования функций из схемы anfun установленной в базе данных пользователь должен обладать правами на использование схемы и исполнение функций внутри схемы. Чтобы предоставить пользователю эти права используется команды:

```
GRANT USAGE ON SCHEMA anfun TO <user role>

GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA anfun TO <user role>
```

6.2. Метод кластеризации - K-means

Первым методом кластеризации, реализованным в рамках работы, стал K-MEANS. Определение функции, реализующей метод выглядит следующим образом:

```
FUNCTION anfun.Kmeans(_tbl, k, p=2.0) RETURNS table(id, X, Y, C)
```

Принимая в качестве параметров таблицу с данными, функция кластеризует их и возвращает таблицу с дополнительным столбцом, в котором определены номера кластеров. Наравне с именем исходной таблицы в функции также передается параметр алгоритма – количество кластеров K и опциональный параметр P, указывающий на модификацию метода расчета расстояний - степенной параметр расстояний Минковского. По умолчанию параметр установлен на значение 2, что соответствует Евклидовой метрике расчета расстояний.

Таблица, передаваемая функции в качестве аргумента, должна содержать 3 столбца: уникальные целые значения точек, и соответственно по две координаты для каждой точки. Используя эти данные, функция реализует алгоритм кластеризации K-MEANS и возвращает таблицу с четырьмя столбцами, такую, что значения первых трех столбцов совпадают с исходной таблицей с точностью до перестановки строк, а четвертый содержит номера кластеров, которым принадлежат соответствующие точки.

Пример вызова функции (а также пример обращения к функции генерации данных random2DimDotsBlobs) и соответствующий результат, возвращаемый ею представлены ниже.

```
>> IN

SELECT * FROM anfun.KMEANS('anfun.random2DimDotsBlobs(100,5,0.6)', 5);

>> OUT

------------------------------------------------------------

|  id   |         X          |          Y         |    C    |
------------------------------------------------------------
|   1   | -0.886080343466947 | -0.100481258086202 |    3    |
|   2   | 0.127891907945575  | -0.970856501812635 |    3    |
|  ...  |        ...         |         ...        |   ...   |
|  100  | -1.88230500926938  | -0.457830247602034 |    3    |

------------------------------------------------------------

Total running time of the script: ( 0 minutes 0.438 seconds)
```

6.3. Метод кластеризации – K-medians

Еще одним реализованным методом стал алгоритм K-medians. Определение функции, реализующей метод выглядит следующим образом:

```
FUNCTION anfun.Kmedians(_tbl, k, p=2.0) RETURNS table(id, X, Y, C)
```

Принимая в качестве параметров таблицу с данными, функция кластеризует их и возвращает таблицу с дополнительным столбцом, в котором определены номера кластеров. Помимо имени исходной таблицы в функции также передается параметр алгоритма k – количество кластеров, и опциональный параметр P, указывающий на модификацию метода расчета расстояний - степенной параметр расстояний Минковского. По умолчанию параметр установлен на значение 2, что соответствует Евклидовой метрике расчета расстояний.

Таблица, передаваемая функции в качестве аргумента, должна содержать 3 столбца: уникальные целые значения точек, и соответственно по две координаты для каждой точки. Используя эти данные, функция реализует алгоритм кластеризации K-medians и возвращает таблицу с четырьмя столбцами, такую, что значения первых трех столбцов совпадают с исходной таблицей с точностью до перестановки строк, а четвертый содержит номера кластеров, которым принадлежат соответствующие точки.

Пример вызова функции (а также пример обращения к функции генерации данных random2DimDotsBlobs) и соответствующий результат, возвращаемый ею представлены ниже.

```
>> IN

SELECT * FROM anfun.Kmedians('anfun.random2DimDotsBlobs(400,5,0.6)',5);

>> OUT

------------------------------------------------------------

|  id   |         X          |          Y         |    C    |
------------------------------------------------------------
|   1   | 0.390280675761167  | 0.390280675761167  |    1    |
|   2   | 0.389860739333647  | 0.553050599698853  |    1    |
|  ...  |        ...         |         ...        |   ...   |
|  2000 | 0.956120890950159  | 0.418550760601775  |    5    |

------------------------------------------------------------

Total running time of the script: ( 0 minutes 0.451 seconds)
```

6.4. Метод кластеризации – DBSCAN

Следующим реализованным методом стал алгоритм кластеризации DBScan. Определение функции, реализующей метод выглядит следующим образом:

```
FUNCTION anfun.DBSCAN(_tbl, n, e, p=2.0) RETURNS table(id, X, Y, C)
```

Принимая в качестве параметров таблицу с данными, функция кластеризует их и возвращает таблицу с дополнительным столбцом, в котором определены номера кластеров. Наравне с таблицей с данными, функция также принимает значение параметра n, который характеризуют количество соседей, а также значение параметра e, задающего радиус окрестности точки. Последним значением, передаваемым в функцию, является опциональный параметр P, принцип работы которого идентичен описанному в K-MEANS.

Таблица, передаваемая функции в качестве аргумента, должна содержать 3 столбца: уникальные целые значения точек, и соответственно по две координаты для каждой точки. Используя эти данные, функция реализует алгоритм кластеризации DBScan и возвращает таблицу с четырьмя столбцами, такую, что значения первых трех столбцов совпадают с исходной таблицей с точностью до перестановки строк, а четвертый содержит номера кластеров, которым принадлежат соответствующие точки.

Пример вызова функции (а также пример обращения к функции генерации данных random2DimDotsCircles) и соответствующий результат, возвращаемый ею представлены ниже.

```
>> IN

SELECT * FROM anfun.DBSCAN('anfun.random2DimDotsCircles(50,2,1)', 3, 0.5)

>> OUT

------------------------------------------------------------

|  id   |         X        |          Y        |    C     |
------------------------------------------------------------
|   66  | 1.21016401631128 | -1.66603106278408 |    -1    |
|   82  | 1.79869858114246 |  1.79869858114246 |    -1    |
|  ...  |        ...       |        ...        |   ...    |
|  100  | 1.17477098984541 |  1.79869858114246 |    52    |

------------------------------------------------------------

Total running time of the script: ( 0 minutes 2.140 seconds)
```

6.5. Метод кластеризации FOREL

Следующим реализованным методом стал алгоритм кластеризации DBScan. Определение функции, реализующей метод выглядит следующим образом:

```
FUNCTION anfun.FOREL(_tbl, r, p=2.0) RETURNS table(id, X, Y, C)
```

Принимая в качестве параметров таблицу с данными, функция кластеризует их и возвращает таблицу с дополнительным столбцом, в котором определены номера кластеров. Наравне с таблицей с данными, функция также принимает значение параметра r – радиус, и опциональный параметр P, указывающий на модификацию метода расчета расстояний - степенной параметр расстояний Минковского. По умолчанию параметр установлен на значение 2, что соответствует Евклидовой метрике расчета расстояний.

Таблица, передаваемая функции в качестве аргумента, должна содержать 3 столбца: уникальные целые значения точек, и соответственно по две координаты для каждой точки. Используя эти данные, функция реализует алгоритм кластеризации FOREL и возвращает таблицу с четырьмя столбцами, такую, что значения первых трех столбцов совпадают с исходной таблицей с точностью до перестановки строк, а четвертый содержит номера кластеров, которым принадлежат соответствующие точки.

Пример вызова функции (а также пример обращения к функции генерации данных random2DimDotsBlobs) и соответствующий результат, возвращаемый ею представлены ниже.

```
>> IN

SELECT * FROM anfun.FOREL('anfun.random2DimDotsBlobs(100,5,0.6)',0.5);

>> OUT

------------------------------------------------------------

|  id   |         X          |          Y         |    C    |
------------------------------------------------------------
|   1   | 0.728952541810954  | 0.882498744245906  |    1    |
|   2   | 0.730103158665772  | 0.881869907475448  |    1    |
|  ...  |        ...         |         ...        |   ...   |
|  2000 | 0.320956238223132  | 0.142824800118549  |    3    |

------------------------------------------------------------

Total running time of the script: ( 0 minutes 0.331 seconds)
```

6.6. Метод кластеризации – Aglomerative

Последним из реализованных методов кластеризации стал Агломеративный алгоритм. Определение функции, реализующей метод выглядит следующим образом:

```
FUNCTION anfun.Aglomerative(_tbl, method=1, p=2.0) RETURNS table(id, X, Y, C)
```

Принимая в качестве параметра таблицу с данными, функция кластеризует их и возвращает таблицу с дополнительным столбцом, в котором определены номера кластеров. Наравне с таблицей функция с таблицей, функция также принимает еще два параметра значение, которых может быть задано на усмотрение пользователем. Один из этих параметров является уже хорошо знакомым параметром P, задающий расстояния Минковского, в то время как параметр method отвечает за способ определения оптимальности разбиения точек на кластеры на каждой итерации работы алгоритма. Значение параметра является целым и может быть равно 1 или 2. Если алгоритм получает значение 1, то для оценки качества кластеризации на каждой итерации он использует метрику Silhouette. Если алгоритм получает значение 2, то для оценки качества кластеризации на каждой итерации он использует метрику Dunn.

Пример вызова функции (а также пример обращения к функции генерации данных random2DimDotsBlobs) и соответствующий результат, возвращаемый ею представлены ниже. В процессе работы алгоритм выводит сообщения о уточнении наилучшей оценки и соответствующего ей результата. Вердикт Conserve сообщает

об ожидании достаточной кластеризованности данных, вердикт skip обозначает отсутствие улучшений на данной итерации, вердикт Update сообщает о наличии улучшения оценки и, как следствие, ее обновление и замены оптимального разбиения данных на кластеры.

```
>> IN

SELECT * FROM anfun.Aglomerative('anfun.random2DimDotsBlobs(20,5,1)', 1, 2)

>> OUT

Notion: 100: 0.6848370284732838 (Conserve)

...

Notion: 5: 0.9382479234675392 (Update)

Notion: 4: 0.8347233402394823 (Skip)

...

Notion: 1: 0.7493482348248324 (Skip)

------------------------------------------------------------

|  id   |         X         |          Y         |    C    |
------------------------------------------------------------
|   1   | 0.221836018815645 | 0.948778059861219 |    1    |
|   21  | 0.598764481757357 | 0.712446860687442 |    21   |
|  ...  |        ...        |        ...        |   ...   |
|   57  | 0.706261731236634 | 0.416147927641157 |    41   |

------------------------------------------------------------

Total running time of the script: ( 0 minutes 31.240 seconds)
```

6.7. Метрика оценки - Silhouette

Первой из реализованных метрик оценки качества кластеризации была метрика Silhouette. Ниже представлено определение функции, выполняющей оценку кластеризованных данных по этой метрике:

```
FUNCTION anfun.Sillhuette(_data, p=2.0) RETURNS numeric
```

Принимая в качестве параметра таблицу с данными содержащую результат кластеризации, возвращает число – значение метрики Silhouette. Эта таблица является результатом работы функций, реализующих методы кластеризации и

содержит 4 столбца, которые описывают идентификатор точки, ее координаты и идентификатор кластера, которому она принадлежит соответственно. Второй параметр - это уже известное значение степенного параметра из формулы расчета расстояния Минковского.

Далее представлен результат вызова и работы функции.

```
>> IN

SELECT * INTO KMC FROM anfun.KMEANS('anfun.random2DimDotsBlobs(100,5,0.6)', 5);

SELECT anfun.Sillhuette('KMC')

>> OUT

0.98712756208706756961518657458291

Total running time of the script: (0 minutes 4.124 seconds)
```

Поскольку метрика Silhouette является относительной, то область значений функции есть множество действительных чисел в диапазоне от -1 до 1.

6.8. Метрика оценки – Dunn

Еще одной метрикой оценки качества кластеризации, реализованных в рамках данной схемы, была метрика Dunn. Ниже представлено определение функции, выполняющей оценку кластеризованных данных по этой метрике:

```
FUNCTION anfun.Dunno(_data, p=2.0) RETURNS numeric
```

По аналогии с метрикой Silhouette функция принимает в качестве параметра таблицу с данными содержащую результат кластеризации и возвращает число – значение метрики Dunn. Эта таблица также является результатом работы функций, реализующих методы кластеризации и содержит 4 столбца, которые описывают идентификатор точки, ее координаты и идентификатор кластера, которому она принадлежит соответственно. Второй параметр - это уже известное значение степенного параметра из формулы расчета расстояния Минковского.

Далее представлен результат вызова и работы функции.

```
>> IN

SELECT * INTO KMC FROM anfun.KMEANS('anfun.random2DimDotsBlobs(100,5,0.6)', 5);

SELECT anfun.Dunno('KMC')

>> OUT

0.6896325487321876520089845786 5416

Total running time of the script: (0 minutes 6.213 seconds)
```

Так как было сказано ранее метрика Dunn в отличии от silhouette является абсолютной метрики, то и результат, получаемый в результате применения ее является действительным числом, принимающим в диапазоне от 0 до +∞.

6.9. Метрика оценки – DB

Заключительной метрикой оценки качества кластеризации, реализованных в рамках данной схемы, была метрика DB. Ниже представлено определение функции, выполняющей оценку кластеризованных данных по этой метрике:

```
FUNCTION anfun.DavidBoulduin(_data, p=2.0) RETURNS numeric
```

Принимая в качестве параметра таблицу с данными содержащую результат кластеризации, возвращает число – значение метрики DB. Эта таблица является результатом работы функций, реализующих методы кластеризации и содержит 4 столбца, которые описывают идентификатор точки, ее координаты и идентификатор кластера, которому она принадлежит соответственно. Второй параметр - это уже известное значение степенного параметра из формулы расчета расстояния Минковского.

Далее представлен результат вызова и работы функции.

```
>> IN

SELECT * INTO KMC FROM anfun.KMEANS('anfun.random2DimDotsBlobs(100,5,0.6)', 5);

SELECT anfun.DavidBoulduin('KMC')

>> OUT

0.6094706577738924415949529431602102175245

Total running time of the script: (0 minutes 0.203 seconds)
```

6.10. Прочие функции

В то числе в рамках схемы были реализованы вспомогательные функции, перечень которых приведен ниже.

- Dots2DimShow – функция предназначенная для внутреннего обращения; по названию таблицы, содержащей данные о координатах точек, возвращает содержимое этой таблицы;
- Dots2DimClustered – функция предназначенная для внутреннего обращения; по названию таблицы, содержащей данные о координатах точек и результатах кластеризации, возвращает содержимое этой таблицы;
- Dist – функция предназначенная для внутреннего обращения; по координатам двух точек и значению степенного параметра возвращает расстояние Минковского между этими точками;
- InitClusters – функция предназначенная для внутреннего обращения; по названию таблицы, содержащей данные о координатах точек, возвращает таблицу с теми же данными, расширенную начальной информацией для проведения кластеризации;
- maxi – функция предназначенная для внутреннего обращения; из двух полученных в качестве аргументов чисел возвращает то, которое является большим;
- mini – функция предназначенная для внутреннего обращения; из двух полученных в качестве аргументов чисел возвращает то, которое является меньшим;
- random2DimDotsCircles – функция по заданным параметрам, генерирующая множество точек, сгущенных на концентрических окружностях разного радиуса;
- random2DimDotsBlobs – функция по заданным параметрам, генерирующая множество точек, сгущенных внутри окружностей равного радиуса, распределенных на плоскости.

В тексте подробнее эти функции рассмотрены не будут, так как они отклоняются от темы работы.

7. Визуализация результатов

Для того, чтобы визуализировать результат работы алгоритма было принято решение представить его в виде точечной диаграммы в Microsoft Excel. Далее приведено подробное описание этой процедуры.

- После применения алгоритма к набору данных необходимо выгрузить полученную таблицу в виде файла с расширением csv.
- Следующим шагом нужно выделить все ячейки полученной таблицы, перейти во вкладку “Данные” и нажать кнопку “Текст по столбцам”
- В появившемся диалоговом окне поставить маркер в поле “с разделителем” и нажать кнопку “далее” и нажать
- Затем в качестве символа разделителя выбрать запятую и нажать кнопку “Далее”
- Следующим шагом нажать кнопку “Подробнее” и убедиться, что разделителем целой и дробной частей является запятая, после чего нажать кнопку “Готово”
- Далее необходимо выделить столбцы содержащие координаты x, y и значение ячейки кластер, после чего нажать кнопку “Сводная таблица” в разделе “Вставка”
- В появившемся листе, содержащем сводную таблицу необходимо поместить значение столбца X в поле “Строки” значение столбца C поместить в поле “Столбцы”, а значение столбца Y поместить в поле “Значения” изменив параметры полей значений на “Среднее”
- Выделить все столбцы получившейся таблицы за исключением поля “Общий итог” и создать новый лист поместив туда скопированные ячейки
- В полученной таблице необходимо удалить значение ячейки A1, после чего во вкладке “Вставка” выбрать пункт “Вставить точечную диаграмму”

Ниже приведены примеры визуализации кластеризованных данных на рисунках 2 и 3.

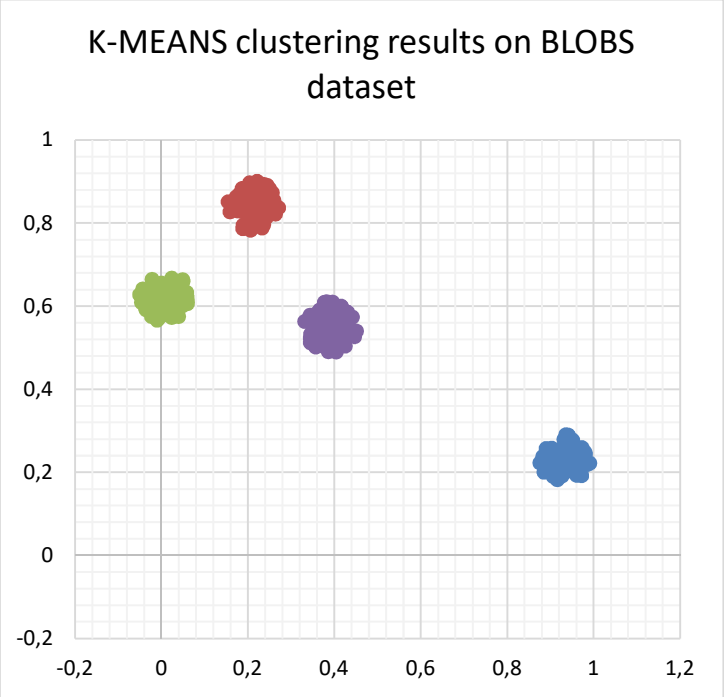

Рисунок 2. Визуализация результатов работы метода K-MEANS на данных полученных с помощью random2DimDotsBlobs

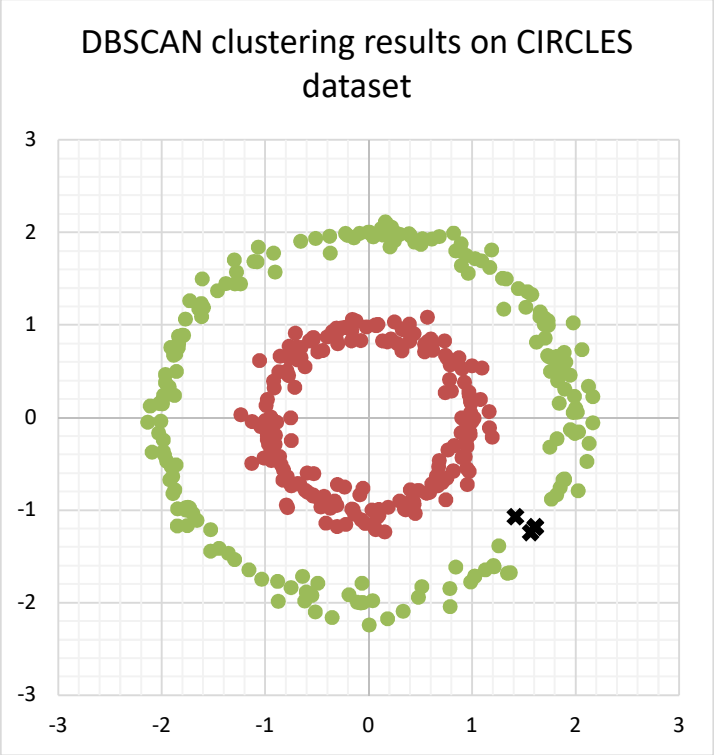

Рисунок 3. Визуализация результатов работы метода DBSCAN на данных полученных с помощью random2DimDotsCircles

8. Заключение

В рамках данной работы был произведен обзор наиболее популярных и полезных алгоритмов кластеризации, а также способов оценки их качества.

Позднее на языке PL/pgSQL был разработан пакет аналитических функций, разворачиваемый в СУБД PostgreSQL и, включающий в себя наравне с другими функции, реализующие такие алгоритмы как: K-means, K-medians, DBScan, Агломеративный алгоритм кластеризации, FOREL а также алгоритмы оценки качества кластеризации по метрикам Silhouette, Dunn и DB.

Результат разработки был опубликован на портале GitHub и доступен по ссылке https://github.com/Twicelab/anfun. К коду приложены файлы спецификации и инструкции по установке. Метод разработки продукта позволяет масштабировать его и добавлять функциональность в будущем, расширяя возможности и оптимизируя его работу.

9. Список литературы

[1] Apache MADlib documentation – URL: http://madlib.apache.org/documentation.html
[2] PostGIS documentation – URL: https://postgis.net/documentation/
[3] PL/R documentation – URL: http://www.joeconway.com/doc/doc.html
[4] PostgreSQL documentation – URL: https://www.postgresql.org/docs/
[5] Md. Sohrab Mahmud, Md. Mostafizer Rahman, Md. Nasim Akhtar, " Improvement of K-means Clustering algorithm with better initial centroids based on weighted average", 7th International Conference on Electrical and Computer Engineering, 2012, pp. 647-650.
[6] Aristidis Likas, Nikos Vlassis, Jakob J. Verbeek, The global k-means clustering algorithm, Pattern Recognition, Volume 36, Issue 2,2003,Pages 451-461
[7] Davoud Moulavi, Pablo A. Jaskowiak, Ricardo J. G. B. Campello, Arthur Zimek, and Jörg Sander, "Density-Based Clustering Validation" Proceedings of the 2014 SIAM International Conference on Data Mining. 2014, 839-847
[8] Загоруйко Н. Г., Ёлкина В. Н., Лбов Г. С. Алгоритмы обнаружения эмпирических закономерностей. — Новосибирск: Наука, 1985.
[9] Lance G. N., Willams W. T. A general theory of classification sorting strategies. 1. hierarchical systems // Comp. J. — 1967. — no. 9. — Pp. 373–380.
[10] Сивоголовко Е. В. Методы оценки качества четкой кластеризации, 2011
[11] Miniakhmetov, Ruslan. (2011). Integrating Fuzzy c-Means Clustering with PostgreSQL. CEUR Workshop Proceedings. 735. 6-10.
[12] K. M. A. Patel and P. Thakral, "The best clustering algorithms in data mining," 2016 International Conference on Communication and Signal Processing (ICCSP), Melmaruvathur, 2016, pp. 2042-2046.
[13] Shifei Ding, L. Zhang and Y. Zhang, "Research on Spectral Clustering algorithms and prospects," 2010 2nd International Conference on Computer Engineering and Technology, Chengdu, 2010, pp. V6-149-V6-153.
[14] Hans-Jürgen Schönig (2018). Mastering PostgreSQL 10
[15] D. L. Davies and D. W. Bouldin, "A Cluster Separation Measure," in *IEEE* Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-1, no. 2, pp. 224-227, April 1979.